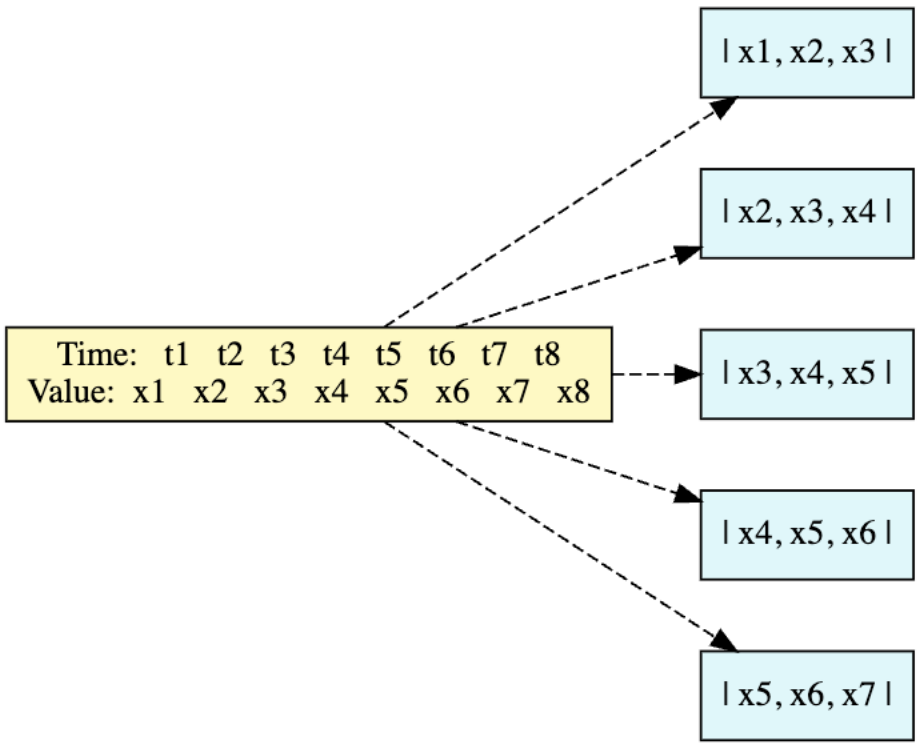
Data Preprocess

Before fit time series into Machine learning or Deep learning model, we need to firstly Transform Data, as shown in figure below.

AutoEncoder is based on **Sequential Data.**



More detail are discussed in [Transform Time Series Data for Supervised Learning: From Sequence to Samples]()

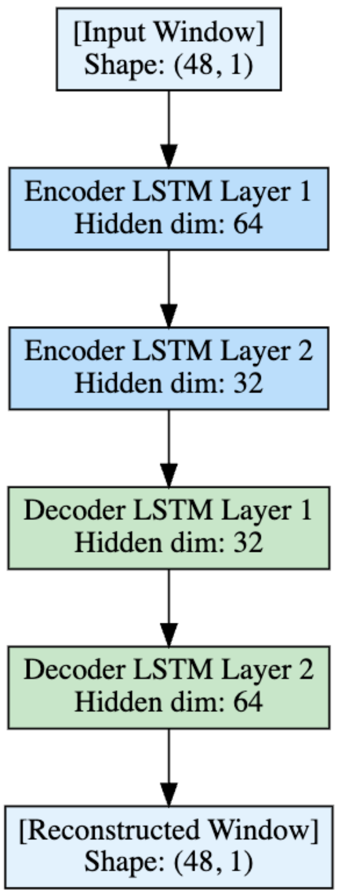
Definition and Architecture of the LSTM Autoencoder

The deep LSTM autoencoder is designed for **sequential** data.

Why LSTM:

LSTMs (Long Short-Term Memory networks) are well-suited for capturing temporal dependencies and sequential patterns within each window.

Architecture:



Encoder: Stacks two LSTM layers that compress the input window into a lower-dimensional latent (embedding) representation.

Hidden dimension: Determines the size of the internal memory for the LSTM at each time step, enabling it to capture complex dependencies.

Embedding dimension: The size of the compressed (encoded) vector; this acts as a "summary" of the input window.

Decoder: Another two LSTM layers, which take the embedding and try to reconstruct the original window.

Two layers for both encoder and decoder: Stacking multiple LSTM layers enables the model to learn more abstract and hierarchical temporal features, improving reconstruction ability and robustness to noise.

Optimization

Loss function: The reconstruction error (usually mean squared error between the original and reconstructed window) is minimized during training, teaching the autoencoder to faithfully reproduce normal patterns.

```
┌─────────────────────┐
│  Input Window       │
│  [x_1, x_2, ..., x_n] │──────────┐
└─────────────────────┘          │
          │                      │
          v                      │
┌─────────────────────────────┐  │
│  LSTM Autoencoder Model     │  │
│ (Encoder + Decoder LSTM)    │  │
└─────────────────────────────┘  │
          │                      │
          v                      │
 [Reconstructed Window]          │
 [ŷ_1, ŷ_2, ..., ŷ_n]            │
          │                      │
          └──────┬───────────────┘
                 v
      Compare Input vs Output
      Calculate Reconstruction Error
        (Mean Squared Error)
                 │
                 v
   ┌─────────────────────────────┐
   │ Backpropagation & Optimizer │
   └─────────────────────────────┘
                 v
      Update Model Weights
                 │
        (Repeat for Each Batch/Epoch)
```

Suppose your **input window** is a batch of three time steps from a univariate series:

Time	Input (x)	Reconstructed ($\hat{y}$)
t_1	10.0	10.2
t_2	11.5	11.2
t_3	10.7	10.8

Calculate Mean Squared Error (MSE):

$$
\begin{aligned}
\text{MSE} &= \frac{1}{3}\left[(10.0 - 10.2)^2 + (11.5 - 11.2)^2 + (10.7 - 10.8)^2\right] \\
&= \frac{1}{3}\left[0.04 + 0.09 + 0.01\right] = \frac{1}{3} \times 0.14 = 0.0467
\end{aligned}
$$

The model will use this MSE to update its weights so the next reconstruction is even closer to the input window.
